

Get Started with DAAL Spark samples on Databricks

This is a getting started guide to DAAL-Spark on Databricks. At the end of this guide, the reader will be able to run a DAAL Apache Spark sample application that runs on Databricks.

Step 1 Start a Databricks cluster

Create a Databricks cluster ([Clusters](#) -> + [Create Cluster](#))

Step 2 run initialization notebook

After you start a Databricks cluster, create a python initialization notebook

Init notebook

Step 2.1

Download latest available daal release from github

```
%sh

url="https://github.com/intel/daal/releases/download/2019_u4/l_daal_oss_p_2019.4.007.tgz"
name="l_daal_oss_p_2019.4.007"
wget $url
tar -xzf "${name}.tgz"
```

Step 2.2

Put all necessary sources into appropriate directory

```
%fs

cp
file:/databricks/driver/l_daal_oss_p_2019.4.007/daal_prebuild/linux/daal/lib/intel64_lin/libJavaAPI.
```

```
%fs

cp
file:/databricks/driver/l_daal_oss_p_2019.4.007/daal_prebuild/linux/tbb/lib/intel64_lin/gcc4.4/libtbb.so.2 /FileStore/jars/
```

```
%fs
```

```
cp
```

```
file:/databricks/driver/l_daal_oss_p_2019.4.007/daal_prebuild/linux/tbb/lib/intel64_lin/gcc4.4/libtbbmalloc.so.2 /FileStore/jars/
```

```
%fs
```

```
cp file:/databricks/driver/l_daal_oss_p_2019.4.007/daal_prebuild/linux/daal/lib/daal.jar /FileStore/jars/
```

Step 2.3

Create a directory for init script and create init script itself.

```
dbutils.fs.mkdirs("dbfs:/databricks/init_scripts/")
dbutils.fs.put("/databricks/init_scripts/init.sh", ""
#!/bin/bash

sudo cp /dbfs/FileStore/jars/daal.jar /databricks/jars/daal.jar

sudo cp /dbfs/FileStore/jars/libJavaAPI.so /databricks/jars/libJavaAPI.so

sudo cp /dbfs/FileStore/jars/libtbbmalloc.so.2 /databricks/jars/libtbbmalloc.so.2

sudo cp /dbfs/FileStore/jars/libtbb.so.2 /databricks/jars/libtbb.so.2

""", True)
```

Step 3 compile DAAL Sample scala code

Information how to do this can be found in the instructions on how to run dual spark samples. the only difference is that we only need the {Sample}.scala file. for example, only KMeans.scala. You can do this with the following command:

```
scalac -d daalkmeans.jar KMeans.scala
```

Step 4 install all libraries

1. Go to "Libraries" tab under your cluster and install dbfs:/FileStore/jars/daal.jar in your cluster by selecting the "DBFS" option for installing jars
2. Install daalkmeans.jar by uploading daalkmeans.jar to DBFS. Select install new and drop daalkmeans.jar into to the appeared window.

Uninstall

Install New

	Name	Type	Status	Source
	daal.jar	JAR	-	dbfs:/FileStore/jars/daal.jar
	daalkmeans	JAR	-	dbfs:/FileStore/jars/34d24c79_645f_4bbc_99d1_e647e55b2586-daalkmeans.jar

Step 5 edit cluster

1. Edit your cluster, adding an initialization script from dbfs:/databricks/init_scripts/init.sh in the "Advanced Options" under "Init Scripts" tab

Advanced Options

Azure Data Lake Storage Credential Passthrough

☐ Enable credential passthrough for user-level data access

Spark

Tags

Logging

Init Scripts

JDBC/ODBC

Permissions

Init Scripts

Type	File Path	Region
DBFS	dbfs:/databricks/init_scripts/init.sh	

2. Add LD_LIBRARY_PATH to environment

Spark

Tags

Logging

Init Scripts

JDBC/ODBC

Permissions

Spark Config

spark.databricks.delta.preview.enabled true

Environment Variables

LD_LIBRARY_PATH=/databricks/jars

PYSPARK_PYTHON=/databricks/python3/bin/python3

3. Reboot the cluster

Step 6 Add test dataset to dbfs

Create New Table

Data source ?

Upload File

DBFS

Other Data Sources

Upload to DBFS ?

/FileStore/tables/ (optional)

Select

File ?

Drop files to upload, or [browse](#).

Step 7 create and run new scala notebook

DAAL KMeans Sark Sample scala notebook

Step 7.1

Import all necessary packages

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

//import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import daal_for_mllib.{KMeans, DAALKMeansModel => KMeansModel}
import org.apache.spark.storage.StorageLevel
```

Step 7.2

Read test dataset

```
val sc = SparkContext.getOrCreate()
val file_location = "/FileStore/tables/KMeans.txt"
val data = sc.textFile(file_location)
val dataRDD = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble)))
```

Step 7.3

run DAAL KMeans Spark job

```
val nClusters = 20

val nIterations = 10

val clusters = KMeans.train(dataRDD, nClusters, nIterations, 1, "random")
```

If everything was fine you will see all information about intel daal spark job



```
Cmd 6

1  val clusters = KMeans.train(dataRDD, nClusters, nIterations, 1, "random")

(12) Spark Jobs
  ▶ Job 15  View (Stages: 1/1)
  ▶ Job 16  View (Stages: 1/1)
  ▶ Job 17  View (Stages: 1/1)
  ▶ Job 18  View (Stages: 1/1)
  ▶ Job 19  View (Stages: 1/1)
  ▶ Job 20  View (Stages: 1/1)
  ▶ Job 21  View (Stages: 1/1)
  ▶ Job 22  View (Stages: 1/1)
  ▶ Job 23  View (Stages: 1/1)
  ▶ Job 24  View (Stages: 1/1)
  ▶ Job 25  View (Stages: 1/1)
  ▶ Job 26  View (Stages: 1/1)

clusters: daal_for_mllib.DAALKMeansModel = daal_for_mllib.DAALKMeansModel@7c652dc7
```

Step 7.4

Just to be sure, that everything work correctly let calculate sum squared errors

```
val cost = clusters.computeCost(dataRDD)

println("Sum of squared errors = " + cost)
```



```
Cmd 8

1  val cost = clusters.computeCost(dataRDD)
2  println("Sum of squared errors = " + cost)

(1) Spark Jobs
Sum of squared errors = 3.7121172278663605E7
cost: Double = 3.7121172278663605E7

Command took 0.28 seconds -- by pivovar3al@outlook.com at 5/12/2020, 2:58:48 PM on test
```